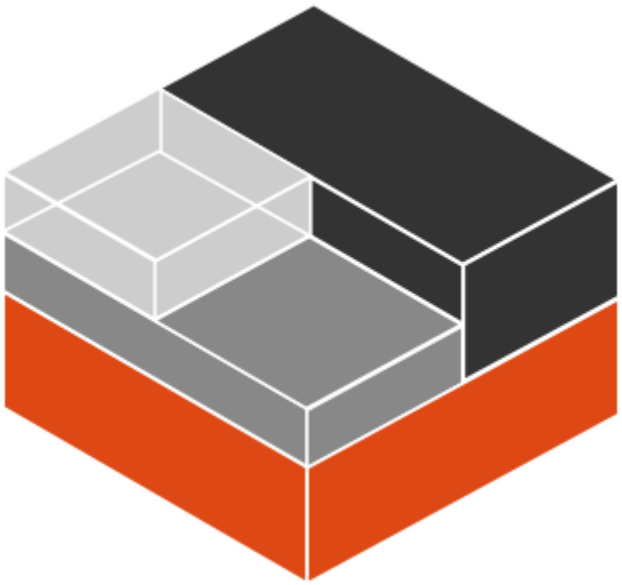
JNOS in an LXC Container



Mark Phillips, NI2O/G7LTT V1.1

Contents:

Abstract:

This document aims to explain the installation of the JNOS Router/BBS for amateur packet radio use. It does not aim to teach you how to use JNOS or configure any of the services. Simply getting a JNOS system functioning is the goal.

Assumptions:

Functioning Proxmox (or other LXC) server.
Familiarity with Proxmox provisioning.
Familiarity with compiling Linux software.
Familiarity with JNOS (DOS or Linux).
Ability to edit Linux OS files.
Functioning connection to the 44net.

Install base LXC:

Prepare a new LXC container with the following properties:

1) Debian 13 (or latest)
2) 4GB disc space. 128MB RAM (yes that's MB not GB)
3) 1 CPU (JNOS is not multi processor capable)
4) Unprivileged
5) Nesting
6) NIC connecting to an ampr/44net LAN with static IP address

Prepare for JNOS install:

JNOS is a router/BBS application that was designed to route RF based data of different protocols between RF LAN's. As such it has no native ethernet abilities and so is unable to communicate with your home LAN. Indeed, its initial creation was long before Ethernet networks were commonly found in homes. We need to create a way in which we can talk to our eventual JNOS installation over our home network.

If your container is running, shut it off. Not the entire Proxmox machine. Just the JNOS container.

Using the Promox server's shell environment, edit the file that starts/configures the new JNOS container every time you start the machine. It is usually found at /etc/pve/nodes/pve-mini/lxc/xxx.conf where xxx is the number of the container itself.

```
nano /etc/pve/nodes/pve-mini/lxc/128.conf
```

Paste the below lines into the bottom of the config file. Then close the file saving your edit on the way out.

```
lxc.cgroup2.devices.allow: c 10:200 rwm
lxc.mount.entry: /dev/net dev/net none bind,create=dir
```

What we did here was add a"tunnel" interface ability to your container. Ordinarily it would not be able to create a tunnel. We need this tunnel so that we can pass data to and from the JNOS instance we'll be building shortly.

Now might also be a good time to add any USB devices such as modems etc to the config file so that the devices get passed from the Proxmox server they are plugged into through to the container.

Start the JNOS container. Assuming it all went well you should be looking at the login prompt for the container. Log in. If the container failed to run, double check the above entries in the config file.

Update the operating system to the latest revisions.

Updating the OS to the latest revisions will require a functioning network. Ensure your network connection within the container is functioning before moving on.

```
apt update && apt upgrade -y
```

Add required libraries & software packages:

```
apt install net-utils build-essential rsync ssh -y
apt install build-essential net-tools traceroute wireguard-dkms screen ntp snmpd -y
apt install daemontools linuxlogo mc tcpdump psmisc nmap mtr wget -y
apt-get install usbutils libncurses5-dev libncursesw5-dev -y
apt-get install libssl-dev -y
```

Create directories required by JNOS.

```
mkdir /root/JNOS
```

Create tunnel scripts:

The tunnel script creates a software tunnel device that will allow the Debian container and the JNOS software to talk to each other. It behaves just like a wired network connection in that it forwards/receives IP based data on a “network device”. In turn both the tunnel device and the JNOS will send this IP data in the correct way. Run the below commands to download a copy of my maketun.sh script. Edit the script with your IP settings and then execute the script. You will find that you now have an additional network interface listed in the Debian container. This interface will not be reflected in the containers settings within Proxmox as it is transient.

```
cd /root/JNOS
wget https://ni2o.ampr.org/documents/JNOS/maketun.sh
```

Edit this script changing the IP addresses where relevant at the beginning of the script.

```
nano /root/JNOS/maketun.sh
```

Make the script executable and then run it.

```
chmod +x /root/JNOS/maketun.sh
/root/JNOS/maketun.sh
```

Now check to see if there’s a new interface called “tun4”

```
Ifconfig
```

Install JNOS:

JNOS instructions are very well maintained by Maiko, VE4KLM and are in active development still today (some 30 years after its initial creation!!). They can be found here https://www.langelaar.net/radio/ve4klm/jnos2/dlcompinst/ and really should be followed!

For a rough and ready install do the following

```
cd /root/JNOS
mkdir src
cd src
rsync -a www.langelaar.net::official .
```
(don't forget the final dot)

When the file sync completes you should have a new directory structure in the current directory. Navigate to the new directory (currently 2.0Q).

Now you must edit the config.h file to turn on all the things you might want JNOS to do.

```
nano config.h
```

Edit the file and then save it.

Now we must run the configurator that checks the sanity of our files and also checks to see if we have the required libraries and other applications.

```
./configure
```

If all is good (it will tell you if things are not) you can finally compile the code

```
make -j4
```

Assuming the 'make' completes without error a new binary file called 'jnos' will be created. Move this file to your JNOS directory and make it executable

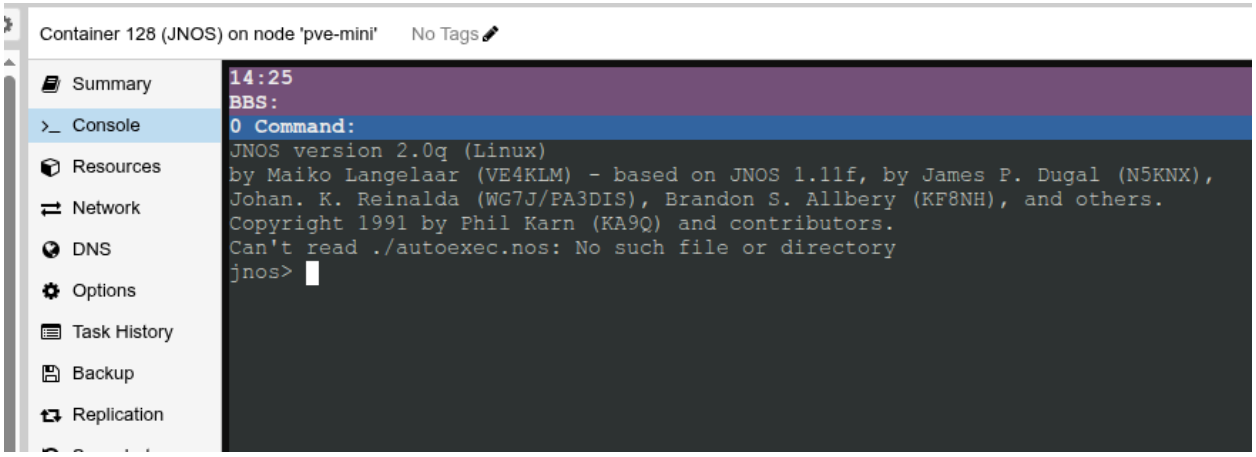
```
mv ./jnos /root/JNOS
chmod +x /root/JNOS/jnos
```

JNOS first run:

Before we do any configuration of the JNOS files lets see if the code actually runs.

```
cd /root/JNOS
./jnos
```

If all went well you should see something like this



Type “exit” to close JNOS and return to Linux.

Configure JNOS:

Did you notice that there was an error message displayed on the screen when we did the first run? JNOS was unable to find any of its config files. If you do an ‘ls’ of the JNOS directory you will see that JNOS created a directory structure for itself. It is here that the config files should have been found. Of course, its a little chicken-n-egg in that the directory structure did not exist before we ran JNOS but JNOS expected to find files in this structure. This problem can now be solved by installing a basic set of config files into the new structure. Follow the below commands to download a “get you going” set of files.

```
cd /root/JNOS
wget https://ni2o.ampr.org/documents/JNOS/autoexec.nos
wget https://ni2o.ampr.org/documents/JNOS/ftpusers
cd /root/JNOS/spool
wget https://ni2o.ampr.org/documents/JNOS/spool/help.tar.gz
wget https://ni2o.ampr.org/documents/JNOS/spool/forward.bbs
wget https://ni2o.ampr.org/documents/JNOS/spool/ftpmotd.txt
tar -xvzf help.tar.gz
```

Edit autoexec.nos:

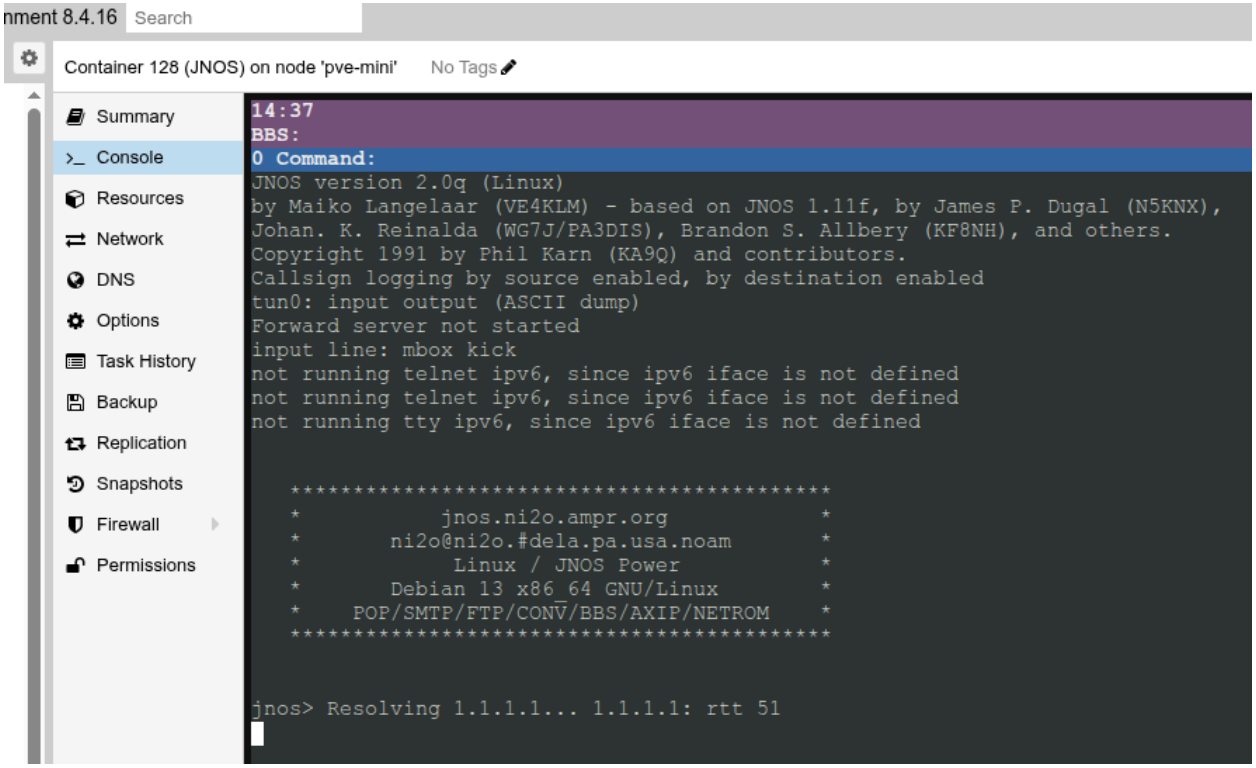
Autoexec.nos is the main setup file for JNOS. A boiler plate file was installed into your system above. Edit this file to reflect your desired system settings. Change your callsign, add modem ports, poll for mail etc etc. Also, edit the ftpusers file. This is where you give permissions to users that log in to the BBS.

```
nano /root/JNOS/autoexec.nos
nano /root/JNOS/ftpusers
```

When you have completed the edits restart the JNOS software

```
cd /root/JNOS/
./jnos
```

You should see a screen similar to the one below. Any errors in your file or misunderstood commands will be shown here.



At this point you should have a working JNOS install. Check it has network connectivity by pinging something like 1.1.1.1. You should then also ping something by name which checvks the DNS is working.

If you simply want an Internet connected BBS/router then we are done. If you want to have RF access on 2 meters then we must add modems and radio's. Refer to "Adding RF ports to JNOS via Proxmox" for more details.